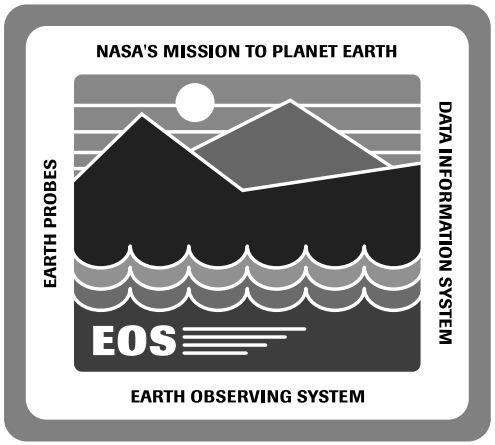
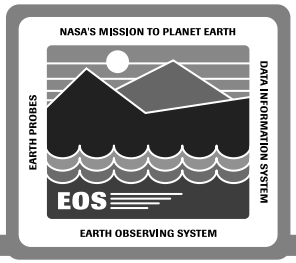


Distributed Objects

Naveen Hota

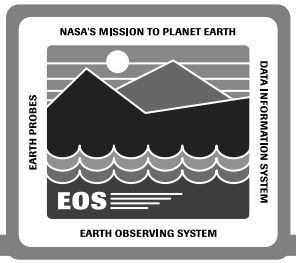
ECS Developers Workshop
30 May 1995

Roadmap



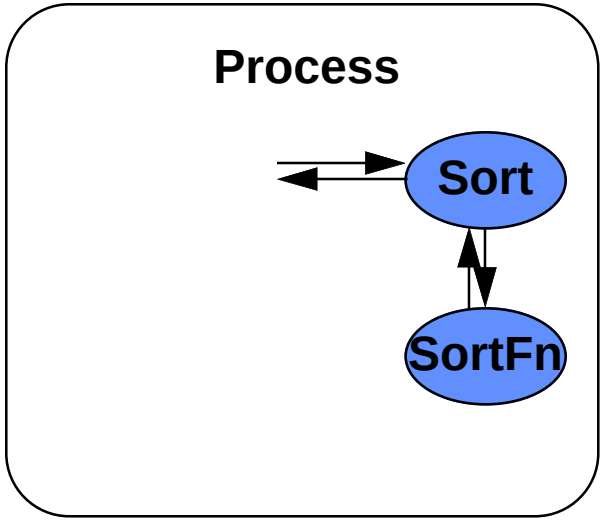
- **Distributed Objects**
- **Special Topics & Insights**
- **Extensions in Security & Directory**
- **An Example: Code walk through**

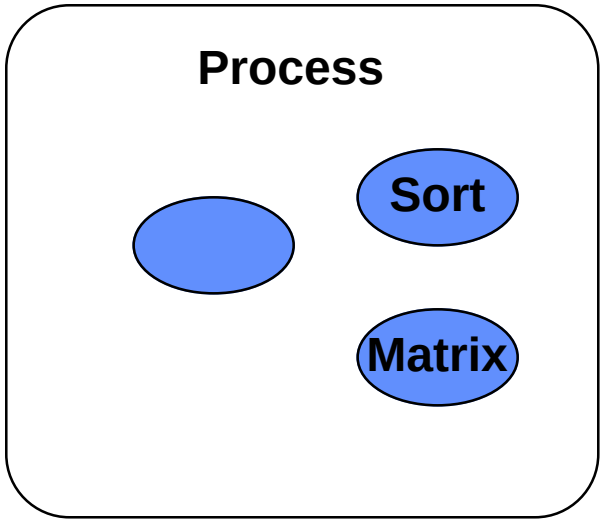
Functional vs. Object Oriented Application Development



Case: Single Address space

- **Functional:**
  - Function oriented
  - No state information
  - Thoroughly familiar
  - Location is a pointer
- **Object Oriented:**
  - Has state and behavior
  - Extendable
  - Provides Encapsulation
  - Inheritance, Abstraction
  - Getting comfortable
  - Location is a pointer





Object vs. Distributed Object

Single address space

Multiple address space

Query server

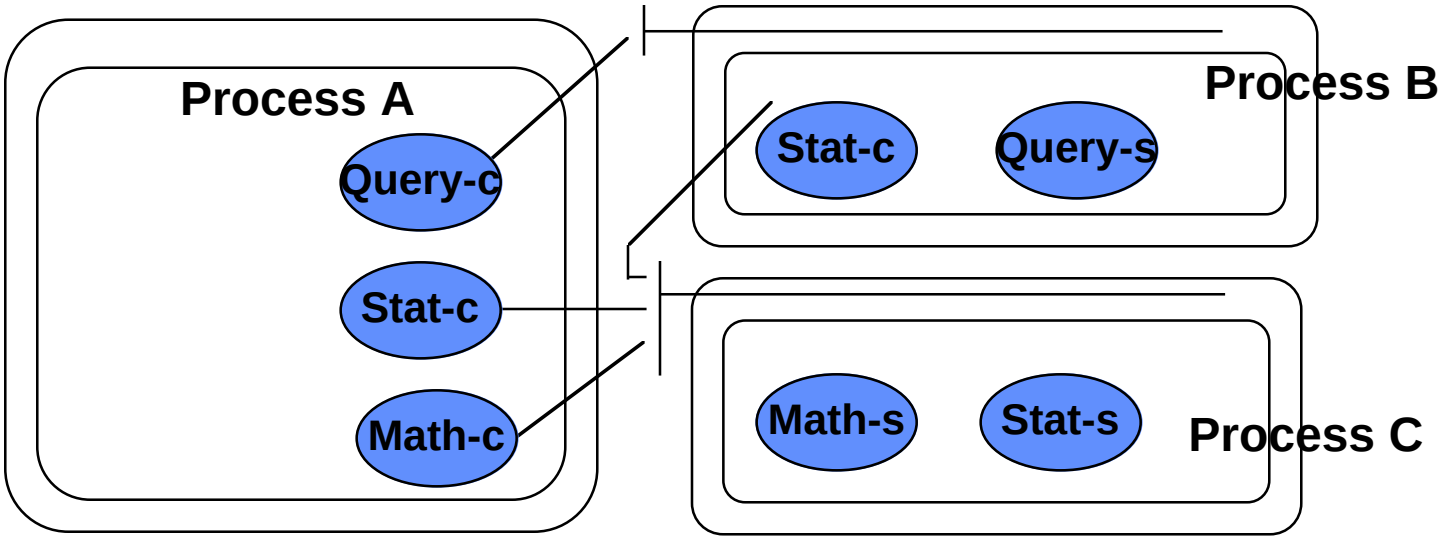
Proxy object

Real object

Client address space

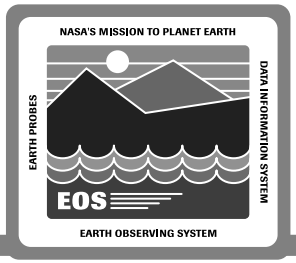
Server address space

Distributed Object Application



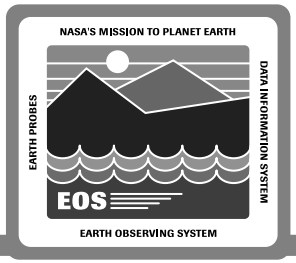
- Multiple address space
- True stateful servers
- State and behavior is captured in objects
- Object location is
  - host, port, transport
  - or interface, object identifiers
  - or a unique name in the namespace

Distributed Objects Application



- **Clear separation between the interface and imlementation**
- **Isolates the application developers from low level communicatio programming**
- **Incremental evolution & flexibility**
- **Location and language independent**
- **Application logic is same irrespective of the physical location of the real object**
- **Share objects**
- **Security may be an issue, but can be dealt with.**
- **Has some overhead.**

Evolution Path



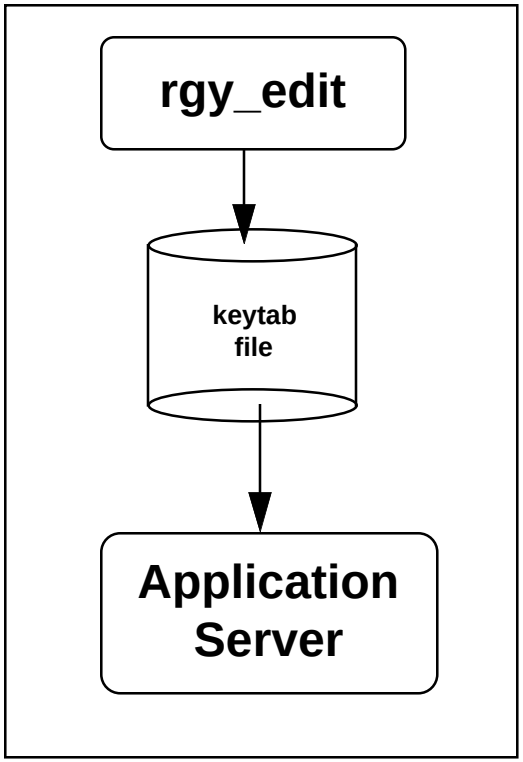
- **End Goal: CORBA (not mature today)**
- **Closest is OODCE**
  - **Same paradigm**
  - **Functional subset of CORBA in DO area**
  - **Has some good additional features: Security, Versioning**
  - **Migration is possible with minimal breakage (but not transparent)**
  - **Level of effort for migration is (relatively) low**

Keytab File

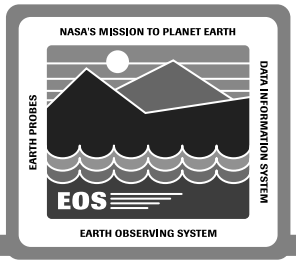
- **Start *rgy_edit* on the desired host by typing “rgy_edit” and do the following:**
  - **rgy_edit =>ktadd -p authSrPrName -f authSrKFile.pwd**
  - **Enter password: ****
  - **Re-enter password to verify: ****
  - **rgy_edit => ktlist -f authSrKFile.pwd**

    /.../baltic.hitc.com/authSrPrName
  - **rgy_edit => quit**

NOTE: Server application must have read access to the Keytab file created

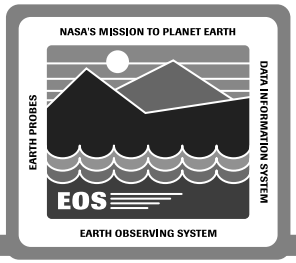


Client/Server Authentication



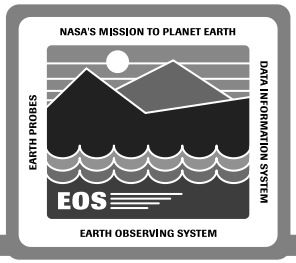
- **For clients and the servers to prove their identities to each other**
- **Server specifies the level of security OFFERED**
  - **Through the SetAuthInfo() API**
  - **Done on a per server basis (NOT on an object/interface basis)**
- **Client specifies the level of security DESIRED**
  - **Through the SetAuthInfo() API**
  - **Done on per object basis**

Client/Server Authentication (continued)



- **SetAuthInfo API**
  - **Authentication protocol (ex: No authentication, DCE secret-key authentication)**
  - **Authorization protocol (ex: Name based authorization, PAC/DCE based authorization)**
  - **protection level (ex: Connect level, Packet_integrity, Packet_privacy, etc.)**
- **For fine grain control at the server a Reference Monitor is used**
  - **Force a client to request certain security levels before servicing its call.**

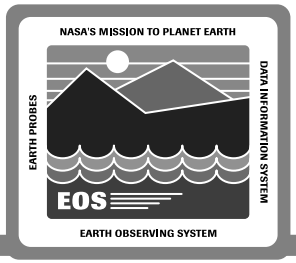
Authorization



- **Checking the privileges to access a resource**
  - **Resource can be operations, files, etc.**
- **Two types of Authorization**
  - **Name based**
    - » **check done based on the name of the principal.**
  - **PAC based**
    - » **check done based on the group to which the principal belongs (extended one)**
    - » **(PAC - Privilege Attribute Certificate, obtained by a principle during Login)**
- **Selected by specifying appropriate authorization protocol in the SetAuthInfo()**

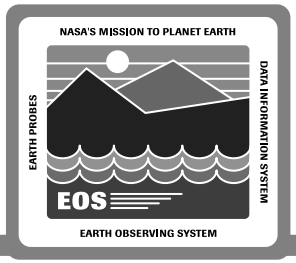
NOTE: Name based Authorization can be used in Legacy applications (database)

ACL Manager



- **Major security component of the server**
- **Provides authorization functionality**
  - **Supports ACL, which has entry/s as follows:**
    - » **entryType:key:permissions (ex: user:joe:rw)**
  - **To check if the client is authorized to access an object/operation/...**
    - » **Reads ACL associate with object and decides privileges**
- **Defines Access Control Permissions**
  - **Application developer definable**
  - **CSS by default creates 32 permissions**
    - » **7 standard permissions - read, write, execute, test, delete, acl-control, insert**
    - » **25 User desired permissions - p1, p2,.....,p25**
- **Creates and associates ACLs to objects**
  - **Stored in memory**

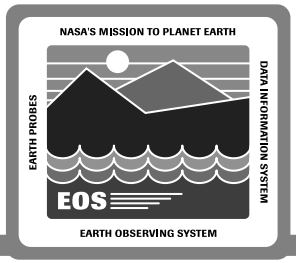
ACL Manager (contd)



- **Creates and manages ACL Databases**
- **Supports standard interface (rdaclif) for external system**
  - **To edit/ manipulate ACLs**
- **CSS provides a class which encapsulates all the above functionalities**

NOTE: CSS will provide ACL Database persistence

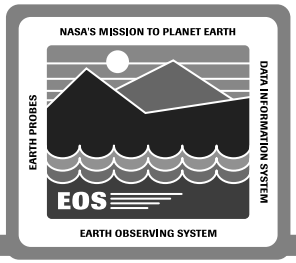
ECSSec Class Functionality



- **ACL schema definition**
  - **defines a set of permissions (valid within an ACL)**
- **ACL manipulation**
  - **through internal interfaces**
  - **through standard external interface (ex: rdacl)**
- **ACL database creation**
  - **maintains a binary tree of ACLs in memory**
- **Authorization checking**
  - **Maps object's ACL with client's PAC using standard DCE ACL checking algorithm**
- **Provides persistent storage of ACLs**
  - **maintains updated ACLs in persistent store**
  - **reads ACLs back into memory when server is restarted**

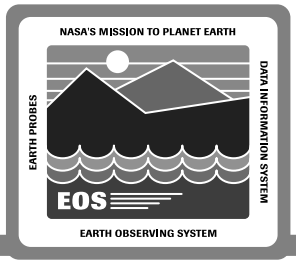
NOTE: Objects not in the same virtual address space should not share ACLs

Threads



- **What are threads ?**
  - **Light weight processes spawned and controlled by a parent**
  - **Each thread shares text and data with the parent**
  - **Each thread can have private and unshared data**
- **Why are threads needed?**
  - **Performance considerations and conceptual clarity**
    - » **for concurrent processing**
    - » **for servers to service multiple client requests**
    - » **for clients to make several requests concurrently**
- **How are threads implemented?**
  - **In the kernel**
  - **In the infrastructure (e.g., DCE)**

Threads (contd.)

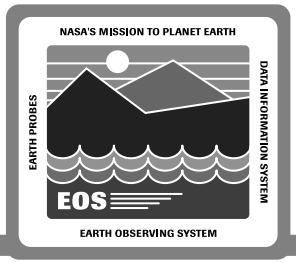


- **Operations on threads available to the parent process:**
  - **Create a thread**
  - **Change the priority**
  - **Change the scheduling policy**
  - **Change run time characteristics**
  - **Wait/kill/join with a thread**

NOTE:

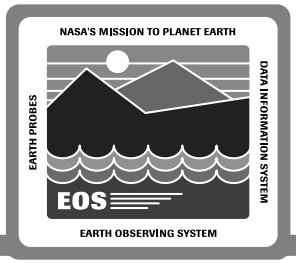
- **Care must be taken to ensure that application code is reentrant (thread safe and usage of thread safe libraries)!!**
- **Limitations: Max number of threads per process is platform dependent**
- **CSS will establish a limit of approximately 500 threads per process**

Time



- **What is Time service?**
  - **Manages clocks on host systems to be synchronized in a network environment**
  - **Provides interface to external time providers to get reliable time**
- **What is provided?**
  - **Application programmers can retrieve time information in various formats**
  - **Application programmers can specify a delta to be applied to simulate clocks**

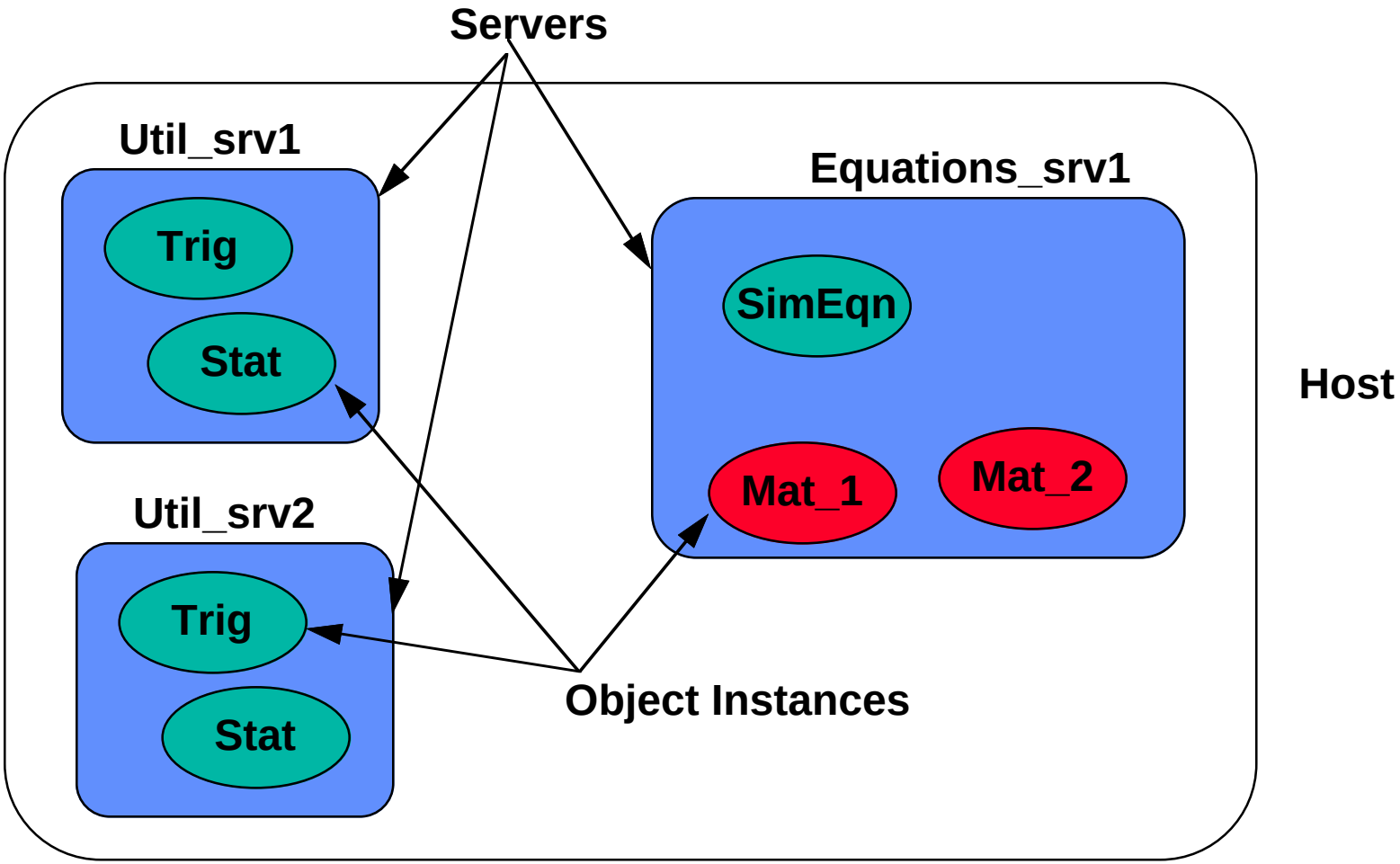
When to Distribute Objects



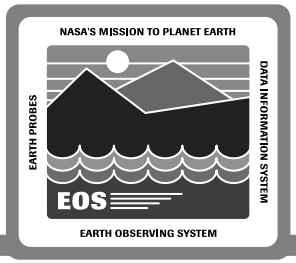
- **When an object needs to be accessed from different processors**
  - **Why?**
    - » **Performance**
    - » **Proximity of data**
    - » **Shared system**
  - **Why not?**
    - » **Network overhead**

- **Make only the needed objects into DO**
- **Identified ~50 DOs (may go up to ~100) for SCDO in Release A of ECS**

Servers & Object Instances



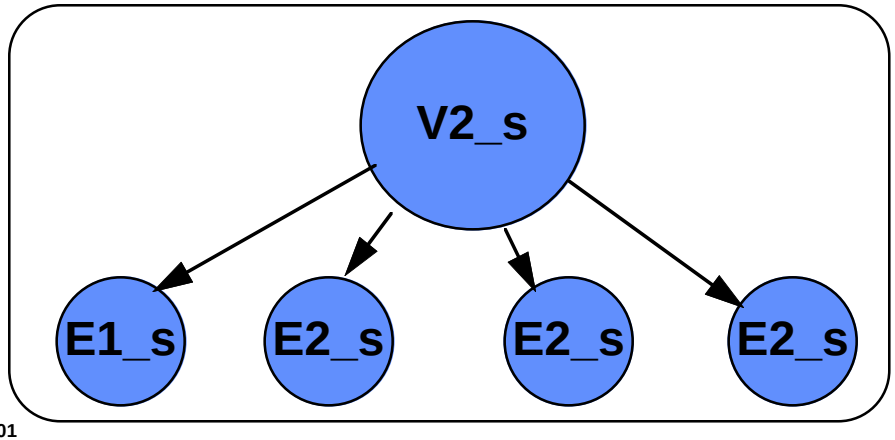
Multiple Servers & Instances



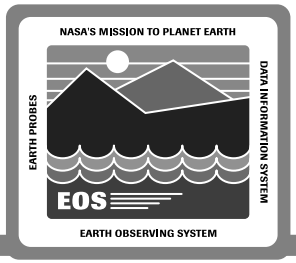
- **Need for Multiple servers**
  - **Performance**
    - » **Load distribution & Effective utilization of resources**
    - » **Locality (reduce network traffic)**
- **Need for multiple Instances in a server**
  - **Multiple clients accessing stateful objects**
- **Interface design**
  - **Conceptual**
  - **Implementation (may need to break it for performance)**

Iterators in DOs

- **What is an iterator**
  - **An OO mechanism to facilitate sequential retrieval of every element of a container class not in the same virtual address space(DOs) as the caller**
- **How should iterators be used**
  - **Provide member functions to retrieve object reference to the next element in a container class**
  - **Client surrogate calls the “next object” method of the container class and rebinds the client object (of the element) with the returned object reference**

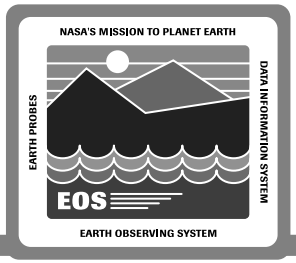


Instantiation of server objects



- **General solution**
  - **Run the server objects continuously**
- **Drawback**
  - **Wasteful of host resources**
- **Alternate solutions**
  - **Factory**
  - **Activation**
  - **Customizing client stubs with the above methods**

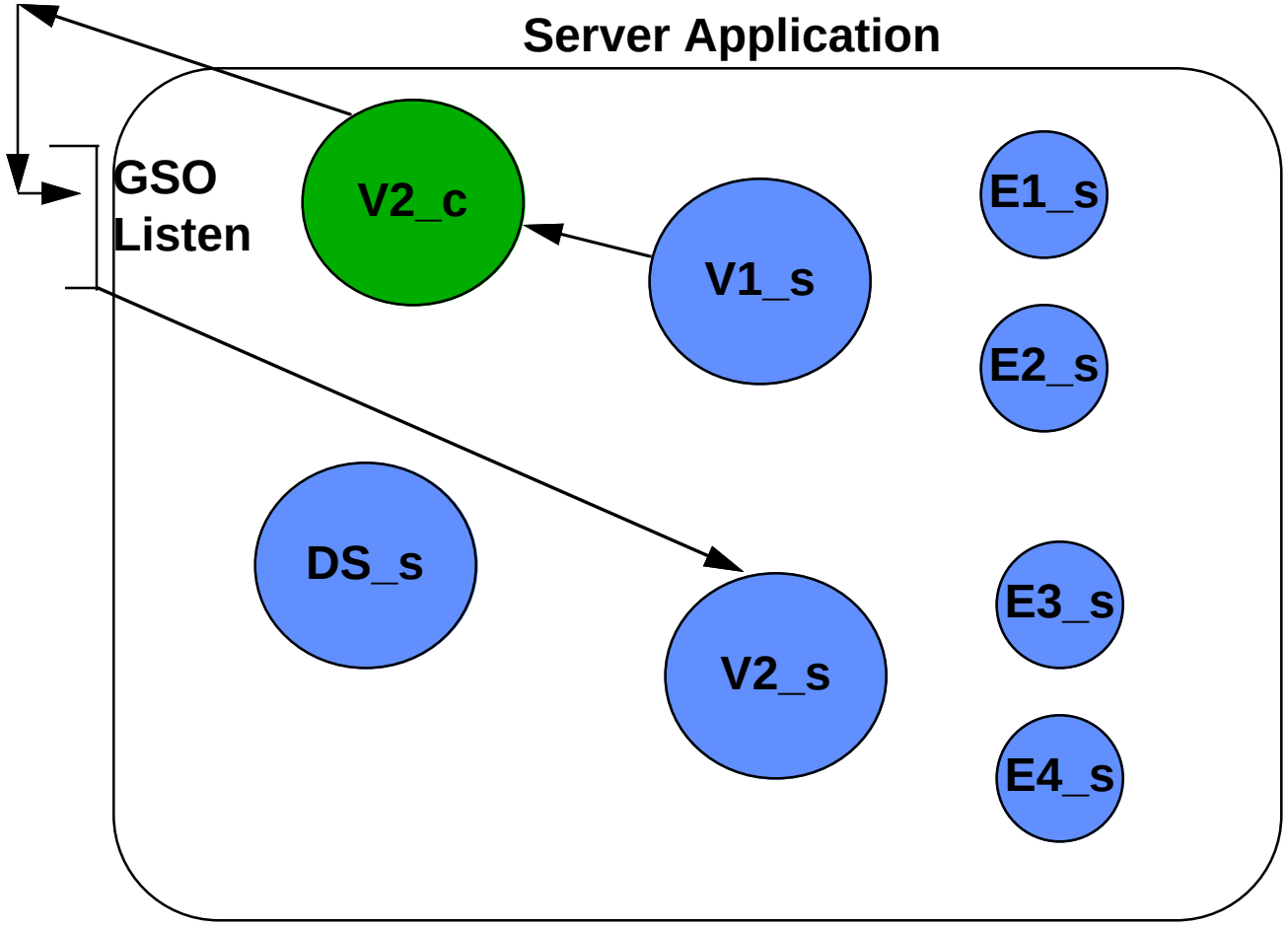
Factory



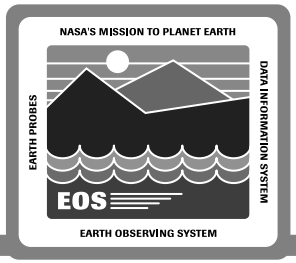
- **How is it done ?**
  - **Maintain another object in the same virtual address space with the sole purpose of creating a server class on demand.**
  - **When instantiation is required, bind to the factory object and request a server object be instantiated.**
  - **The factory returns a reference to the caller for binding to the instantiated server object**
- **Drawback**
  - **Interaction between the created object and other objects with in the server do not take advantage of co-residing in the same virtual address space.**
  - **Factories may not be transient objects**

Solution: Embed the factory functionality in a class that needs interaction with the created class

Factories

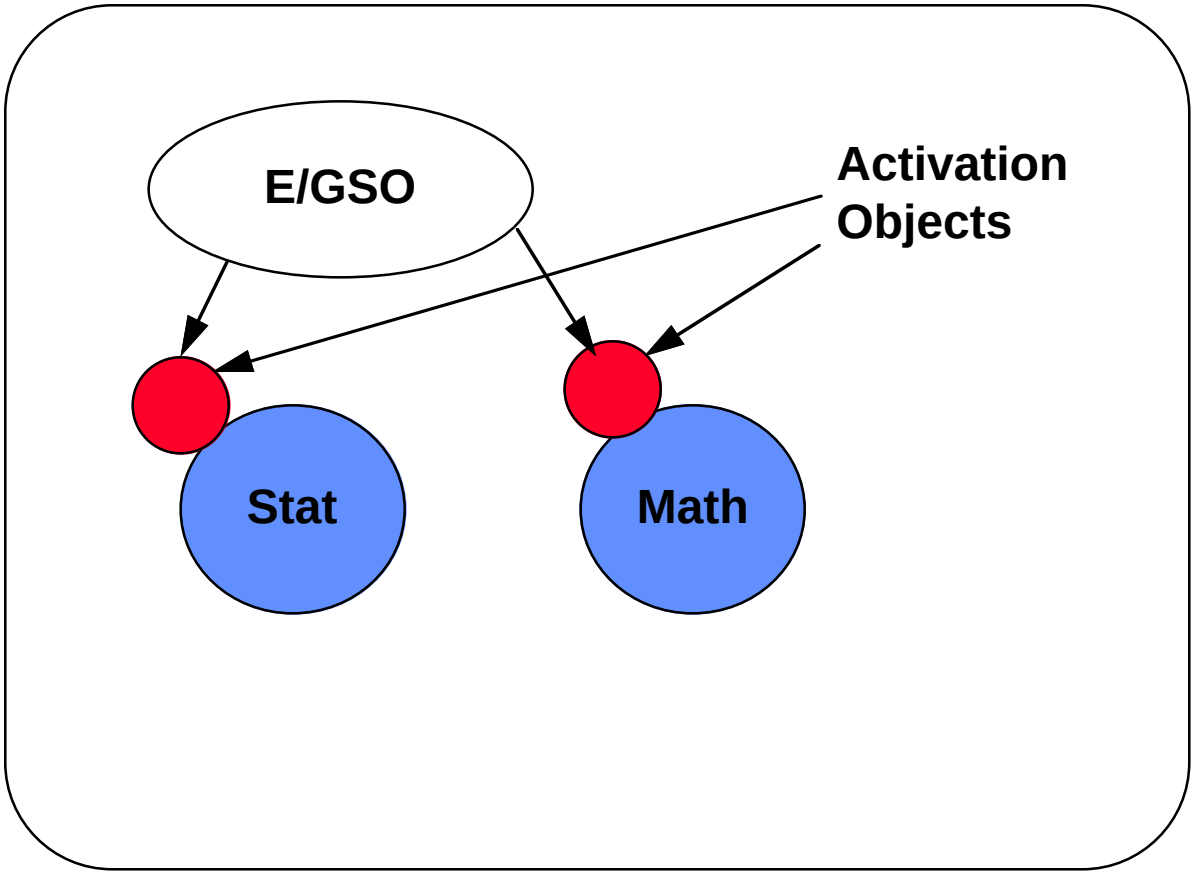

Server Application
GSO
Listen
V2_c
V1_s
E1_s
E2_s
DS_s
V2_s
E3_s
E4_s
Server
Client

Activation

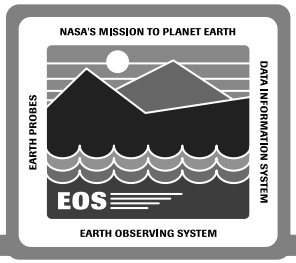


- **How is it done ?**
  - **Create the server object and a corresponding activation object and register them with the GSO (as an activation object)**
  - **Upon receiving an incoming request, GSO instantiates an object, if not already running through the activation object and routes the call to the actual object.**
- **Lookup of a factory is not necessary**
- **Suitable for stateless servers**
- **Drawback**
  - **Can not be used for stateful servers because the GSO does not permit the creation of more than one instance of any given object.**

Activation



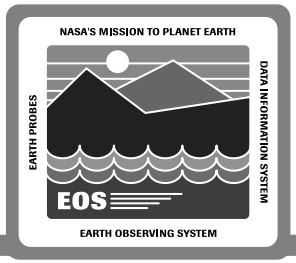
Customizing the Client stubs



- **How is it done ?**
  - **Specialize the base class generated by the IDL compiler for customization and add a new constructor**
  - **Instantiate the client object without a binding**
  - **Instantiate the server object (through a factory) and obtain a reference to it**
  - **Reset the binding to the server object using the reference obtained in the previous step**
- **Migration will be transparent (No factories or lookups in the application code)**

NOTE: Every client object must be specialized from the default class generated by the IDL

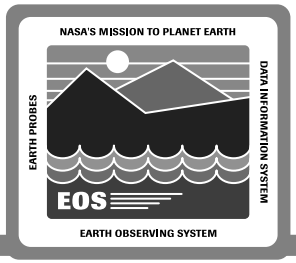
Customizing the Server stubs



- **Why is it needed ?**
  - **IDL doesn't support state information**
  - **Need to add state information to the server object**
  - **May want to add other local functionality (private methods)**
- **How is it done ?**
  - **Specialize the base class generated by IDL compiler for customization.**

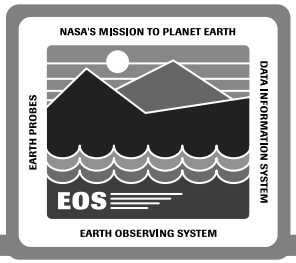
NOTE: Every server object must be specialized from the default class generated by the IDL

Object Passing



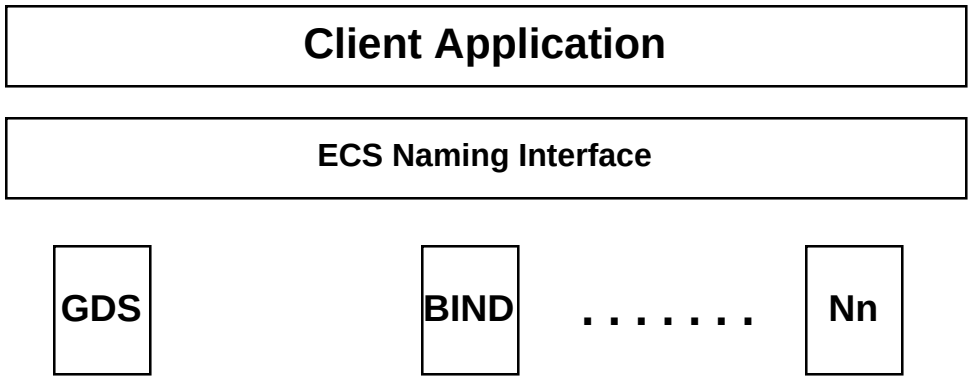
- **Objects can NOT be passed as arguments to Distributed Objects**
  - **Only structures may be passed in OODCE**
- **Alternate ways**
  - **Flatten the state into a byte stream(XDR) and send it and recover it at the other end**
  - **Convert the state into a structure and send it and recover it at the other end**
    - » **Limitation: Converted structure is limited to have only 1 conformant array at the end**
- **Future**
  - **Included in CORBA**
  - **Proposed for DCE1.2 (RFC 48.3 by DEC)**

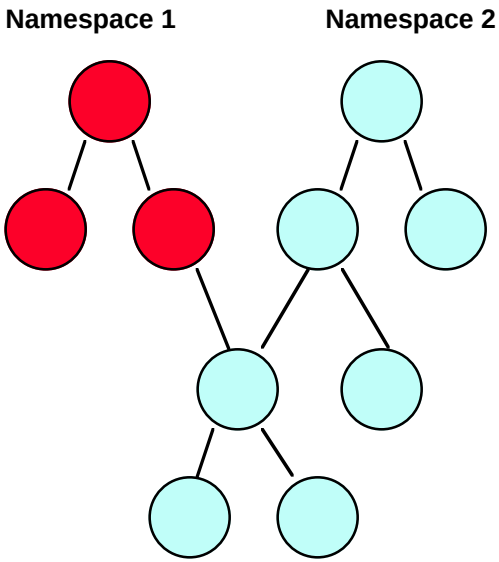
Extending Directory/Naming

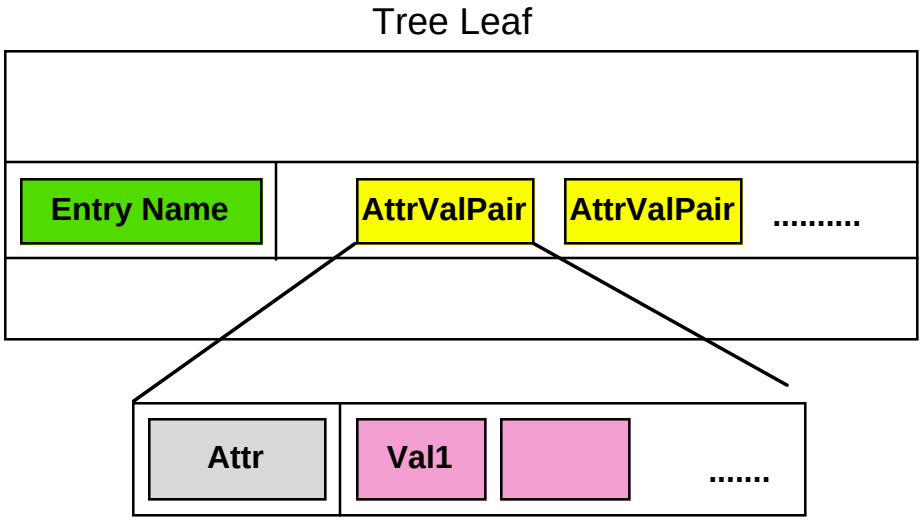


- **Why is it needed?**
  - **It is needed by applications to store application specific state information for other applications to access**
  - **DCE provided interfaces are either too rigid (NSI) or too cumbersome (XOM/XDS).**
- **CSS is providing the capability for extending the directory service through a generic and easy interface**
- **Uses GDS/CDS to store and retrieve user application specific information**
- **Advantages:**
  - **Insulates the application developer from changes in the underlying technology**
  - **Upward compatible with proposed XFN**

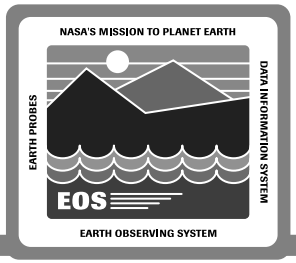
CSS Directory/Naming





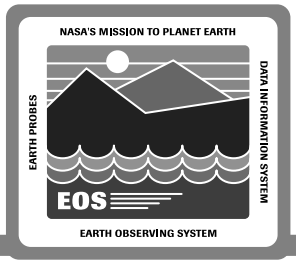


Directory/Naming Classes



- **ECSContext**
  - **Defines the set of bindings with distinct atomic names**
  - **Each context is associated with a namespace**
- **Composite Name**
  - **A nested set of contexts - a complete path**
  - **Intermediary nodes**
    - » **list/read/add sub contexts**
  - **Leaf nodes**
    - » **add/read/delete elements (information associated with each composite name)**
- **Element**
  - **Maintains attribute value pairs**
  - **Provides Add/Delete/Modify/Get attribute value pairs**

Directory/Naming classes (contd.)



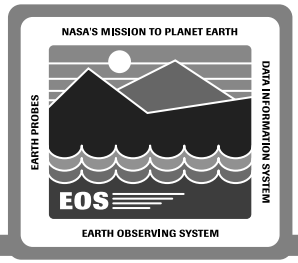
- **ECSAttribute**
  - **Provides functionality to Get/Set attribute name and type**
- **ECSValueList**
  - **A container class of values**
  - **Provides Set/Get/Delete values**
- **ECSValue**
  - **Provides Set/Get a value**

NOTE: It currently uses CDS/GDS to save and retrieve application specific information

Other namespaces can be integrated with the provided layer

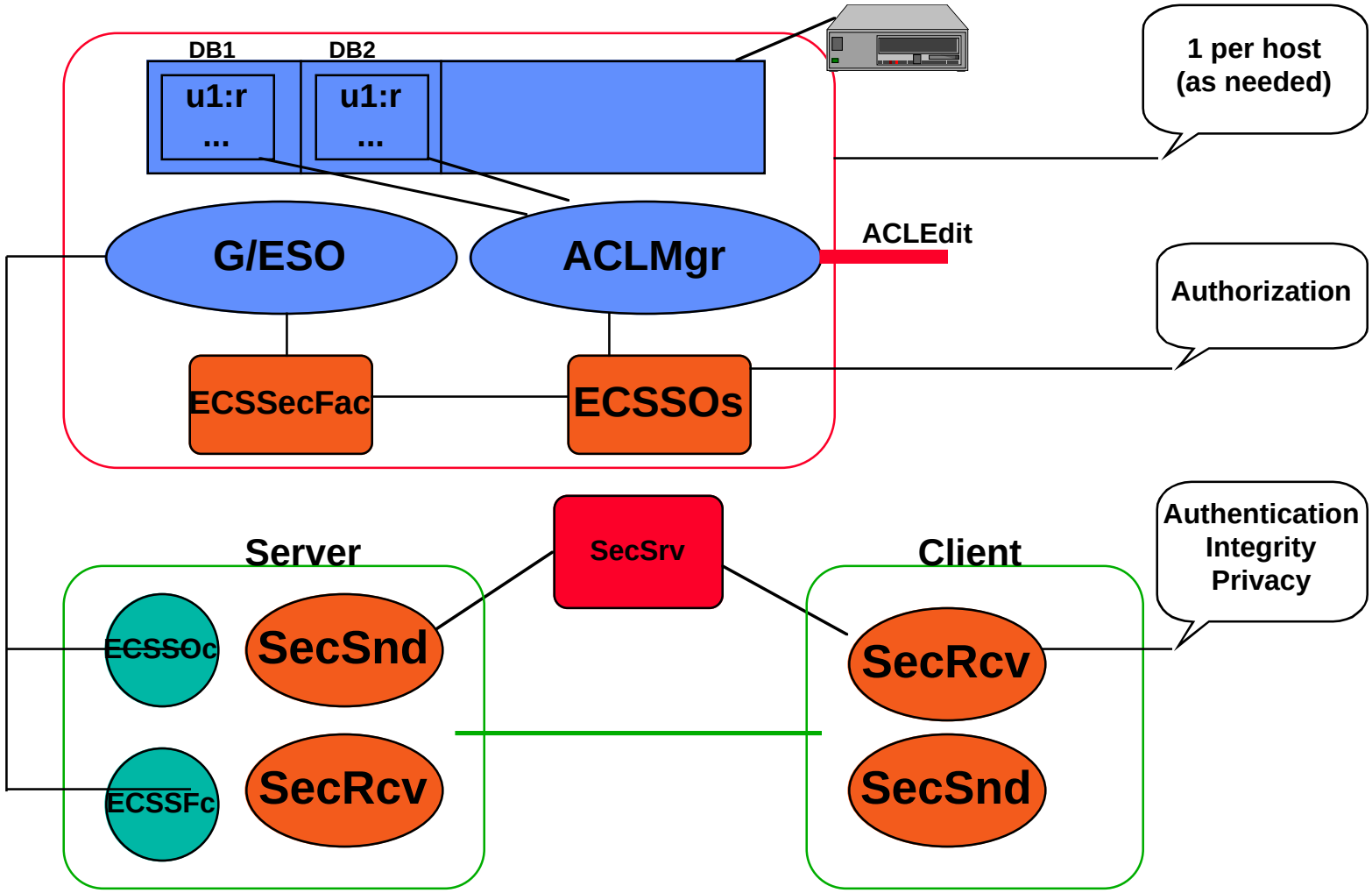
Uses Security present in the native namespace

Security in non RPC Environment

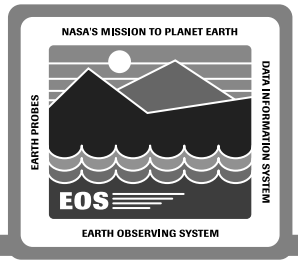


- **The need**
  - **In DCE, security is embedded in RPCs and easy to use by developers**
  - **All ECS applications may not use only DCE RPCs to communicate**
- **What is provided by CSS?**
  - **Same level of security as DCE RPCs**
  - **Requires more involvement on the part of the application developer**

Security in non RPC Environment

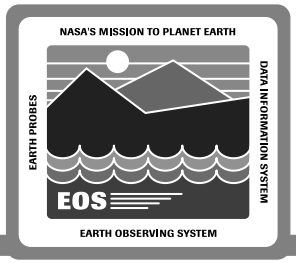


SecSnd Object Functionality

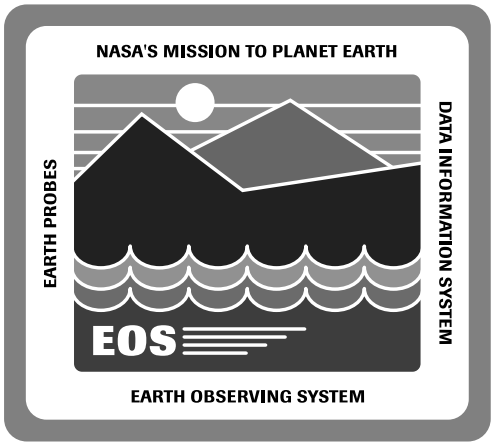


- **AcquireNewCredentials**
  - Get credentials from DCE or renew them
- **SetMsgBody**
  - Set the application message
- **SetMessageFlags**
  - Indicate protection levels
- **SetRecipient**
  - Indicate the recipient identity
- **PrepareMessage**
  - Transforms the message suitable for transfer

SecRcv Object Functionality



- **AcquireNewCredentials**
  - Get credentials from DCE or renew them
- **SetIncomingMsg**
  - Set the received message to be decoded
- **GetMsgFlags**
  - Get protection levels
- **GetSenderIdentity**
  - Get the sender's identity
- **GetMsgBody**
  - Decode the message & verify protections



Code Walk-through

Frank Deluca

30 May 1995

Client/Server Interaction